



**Institute of Information Systems  
Research Group Information Engineering  
University of Bern**

**Technical Report Nr. 130**

**Interfacing SAP R/3  
Logistics with Lotus Notes in  
the ADAM project**

**Michael Röthlin**

**2002-03-03**

The Working Papers of the Institute of Information Systems are preliminary results from current research projects and should initiate scientific discussion; criticism of the content is desired and welcome. Please do not quote without written permission of the authors.

Address: Engehaldenstrasse 8, CH-3012 Bern, Switzerland  
Tel.: ++41 (0)31 631 87 39  
Fax: ++41 (0)31 631 46 82  
E-Mail: [roethlin@ie.iwi.unibe.ch](mailto:roethlin@ie.iwi.unibe.ch)

---

# **Interfacing SAP R/3 Logistics with Lotus Notes in the ADAM project**

*Keywords:*

*Enterprise Application Integration, MRP/ERP systems, Groupware, Data modeling.*

## **Contents**

<b>1</b>	<b>Integrated Enterprise Applications and ADAM.....</b>	<b>1</b>
<b>2</b>	<b>The business problem and requirements behind ADAM.....</b>	<b>2</b>
2.1	The overall business environment and its IT infrastructure.....	2
2.2	The replenishment problem: Components, systems, and project assemblies .....	2
2.3	The materials management consolidation (as-is) process .....	3
2.4	The sales information system (as-is).....	3
2.5	Business requirements for the (to-be) ADAM system .....	4
2.6	Technical requirements for the ADAM system.....	5
<b>3</b>	<b>Technical background.....</b>	<b>6</b>
3.1	SAP R/3 .....	6
3.1.1	<i>The SAP R/3 package .....</i>	<i>6</i>
3.1.2	<i>Client-Server model.....</i>	<i>6</i>
3.1.3	<i>Interfacing SAP R/3 with external systems.....</i>	<i>7</i>
3.1.4	<i>Comparison and selection of RFC and BAPI technologies .....</i>	<i>9</i>
3.2	Lotus Notes .....	10
3.2.1	<i>Product idea .....</i>	<i>10</i>
3.2.2	<i>System components.....</i>	<i>10</i>
3.2.3	<i>Application programming for Lotus Notes.....</i>	<i>11</i>
3.2.4	<i>Interfacing Lotus Notes with ERP systems.....</i>	<i>12</i>
3.2.5	<i>The LotusScript extension library for SAP R/3 (SAP LSX).....</i>	<i>12</i>
3.3	Data access through direct table data retrieval .....	16
<b>4</b>	<b>The ADAM concept and the SAP interfaces .....</b>	<b>18</b>
4.1	ADAM physical system layout .....	18
4.2	Logical system architecture .....	18

---

4.3	ADAM Material subsystem architecture.....	19
4.4	Details of the SAP interface concept .....	20
4.4.1	<i>The ADAM software support modules.....</i>	<i>20</i>
4.4.2	<i>The SAP Connection Manager.....</i>	<i>21</i>
4.4.3	<i>The SAP Query Manager (SAPtabQuery wrapper) .....</i>	<i>21</i>
4.5	Example of data handling in ADAM.....	23
4.5.1	<i>Design of documents containing imported data.....</i>	<i>23</i>
4.5.2	<i>Import filter .....</i>	<i>24</i>
<b>5</b>	<b>SAP LSX: Observations and conclusions .....</b>	<b>25</b>
5.1	The SAP LSX concept and software engineering.....	25
5.2	Maturity and reliability.....	25
5.3	Sensibility to the evolution of software products.....	25
5.4	Data access.....	25
5.5	Conclusions.....	26
<b>Appendix A: Documentation of RFC_READ_TABLE.....</b>		<b>27</b>
A.1	ABAP/4 Code (taken from SAP R/3 repository).....	27
A.2	Import Parameters.....	28
A.3	Export Parameters.....	28
A.4	Tables .....	28
<b>Appendix B: ADAM SAP Interface – SAP table data extraction routines .....</b>		<b>29</b>
<b>Appendix C: The SAPtabQuery function wrapper.....</b>		<b>31</b>
<b>References.....</b>		<b>35</b>

# 1 Integrated Enterprise Applications and *ADAM*

After the "Integrated Enterprise Applications" hype of the last decade which brought a proliferation of new concepts and applications like Enterprise Resource Planning (ERP) systems or Data Warehouses, there still remains a strong demand for systems integration.<sup>1</sup> Despite the expectations, and often due to a lack of resources to master the technical and organizational complexity, many of the so-called integrated solutions are today not able to cover all data processing requirements of an enterprise.

Dedicated applications may make sense in different ways: Whereas transaction oriented environments such as an SAP R/3 system mainly handle highly structured data in online processing and strictly specified processes, document-oriented groupware applications like Lotus Notes are best placed to support the handling of unstructured data in decentralized, flexible workgroup environments.<sup>2</sup> Heterogeneous IT landscapes are (and will remain), despite their conceptual complexity and the need for interfacing, a reality for many enterprises in the years to come.

To cope with this outlook, a class of students in Information Engineering has been put to work on a systems integration problem. Taking a real-world situation in a large manufacturing company, technical and organizational solutions for an interconnection between an MRP/ERP and a groupware system had to be found, the two base systems involved being SAP R/3 on the one and Lotus Notes on the other side. The project code *ADAM* employed stands for "Automation of Data flow from sAles forecast to Materials requirement planning".<sup>3</sup>

This report presents both the business and software engineering problems and depicts some of the technical peculiarities of the design and implementation of the *ADAM* solution. Not considered here are the issues of the development and project management activities in this particular project.<sup>4</sup>

A word to the scope of the *ADAM* approach: The problems and solutions described in this report may apply to a large business community – it was found that 70% of the enterprises using SAP R/3 ERP solutions operate Lotus Notes applications as well.<sup>5</sup>

---

<sup>1</sup> Pender (2000).

<sup>2</sup> Moser (1999), p. 43.

<sup>3</sup> "Automatisierung des Datenflusses zwischen Absatzvorschau und Materialprognose", in German.

<sup>4</sup> These issues are discussed in Röthlin (2001).

<sup>5</sup> Rost (1999), p. 70.

## **2 The business problem and requirements behind *ADAM***

### **2.1 The overall business environment and its IT infrastructure**

The project partner involved in this project is a leading supplier of power electronics products. These products are used in all kind of industrial applications, mainly for power supply purposes. The partner's operation base is Switzerland; the sales, engineering and production line organizations are concentrated on one industrial site, in order to facilitate the overall process control. For the *ADAM* project, and throughout this paper, the business case of the partner company defines the context of the IT solution.

In the mid-1990s, SAP's R/3 system has been chosen as the general application for ERP. Finance and accounting, sales order handling, manufacturing, and specific parts of materials management are today heavily supported by SAP R/3.

On the other side, for collaborative tasks and internal communication, the Lotus Notes groupware application is used as a corporate standard. Hundreds of customized databases have been created to support specific communication and knowledge sharing needs of the diverse business units. Finally, office applications, e.g., spreadsheet processors like Microsoft Excel, have a major importance, especially when it comes to number handling, in order to prepare forecast statements, or for budgeting purposes.

### **2.2 The replenishment problem: Components, systems, and project assemblies**

In order to be able to deliver requested goods to customers on time, the needed quantities of end products must either be (a) taken directly from stock (for standardized modules, i.e., "components"), (b) assembled in lots ("systems"), or (c) built to order ("projects").

As these three ways of delivery may coexist in a given business unit, a shared stock management system has been installed. This facility is called "sales stock", even if some of the modules stored are also built into standard or customer specific systems, i.e., first used for assembly (production) before being shipped as part of a system.

The quantities of components needed to assemble a system are specified with SAP R/3's bills of materials (BOMs). Therefore, an exact description is available on how to derive components requirements if the systems requirements at the upper level are known. The components requirements planning has hence to take into account both the primary requirements (modules sales) and the derived contribution from systems assemblies.

Up to now, the forecast of the resulting components requirements was done using EXCEL spreadsheets. One of the goals of the *ADAM* project was the development of a combined approach, integrating historical data, BOM master data, and sales forecasts for

modules and systems, in order to calculate the components requirements in a collaborative application environment.

## 2.3 The materials management consolidation (as-is) process

Based on the forecast data generated by the different sales units, the central Materials Management office consolidates the materials requirements and enters the results manually into the SAP R/3 material management system. A sample screen of the outcome of this operation is shown in Figure 1 below: The required quantity of material *ES-1300-C* for month 03/2000 is 118 pcs.

Datum	Dispoelement	U-Datum	AM	Zugang/Bedarf	Verfügbare Menge
29.01.00	W-BEST				750
01.02.00	UG-BED	M 02/2000		750-000	044.200
01.03.00	UG-BED	M 03/2000		118	526.200
03.04.00	UG-BED	M 04/2000		150	376.200

Figure 1: Manually entered materials requirements in SAP

## 2.4 The sales information system (as-is)

The sales force operates globally, pursuing sales opportunities with in-house partners and local sales offices, or in collaboration with main contractors. For reporting purposes of the expected order intake, the sales project statistics are collected monthly, in order to give the management a consistent view on current sales activities.

Besides the financial interest behind this data, sales information is also needed to prepare the production and supply system. The information needed to set up such an extended sales statistics is straightforward: A corresponding data set consists typically of a

- Project Name
- System class identifier
- Number of systems contained in project
- Probability that project will be realized and corresponding order is won
- Delivery date
- Sales price.

Due to the collaborative character of the sales pursuit processes, Lotus Notes database applications are used to share common settings, e.g., customer reference or currency codes. Reported data can be collected and processed by all parties concerned, including staff concerned with reporting. Figure 2 shows a sample project list.

Projekt	Technik	Anzahl	Prob. %	Termin	Jan 2000	Feb 2000	Mar 2000	Apr 2000
Gran Canaria	3BTH411414	10	100%	15.01.2000	10	0	0	0
Bora-Bora	3BTH411777	20	50%	01.06.2000	0	0	0	0
Teneriffa II	3BTH414444	11	25%	17.03.2000	0	0	2.75	0
ELBA 3	3BTH415551A	2	80%	11.05.2000	0	0	0	0
Cinque Terre	3BTH411899	9	15%	12.04.2000	0	0	0	1.35
Cordillera	3BTH411123	13	33%	03.01.2000	4.29	0	0	0
Velky Fik	3BTH411546	45	10%	01.08.2000	0	0	0	0
Ko Samui 3	3BTH411123	5	80%	01.04.2000	0	0	0	4
					14.29	0	2.75	5.35

Figure 2: Sales Projects overview

## 2.5 Business requirements for the (to-be) ADAM system

The main task of the system to be designed is to *support Material Requirements Planning, based on input from the Sales Information Systems.*

Figure 3 shows the hierarchy of requirements (projects, systems, and components) as introduced in section 2.2.

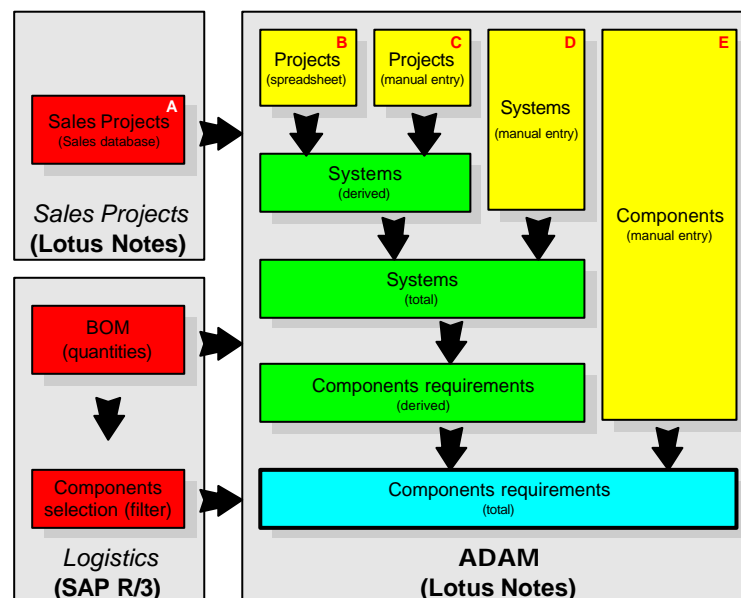


Figure 3: Materials requirements planning with ADAM

Two remarks concerning the input from the SAP system: In order to calculate the requirements for a component (based on the number of systems), the BOM quantities information must be known. Secondly, the materials specifications to be processed are based on the master data contained in the SAP information base. As there may be a lot of non relevant items, filter conditions have to be imposed to prevent undesired data to “congest” the internal ADAM database.

Table 1 summarizes the business requirements for the ADAM system:

ID	Business Requirement
B.1	Simplification, stabilization, and improvement of the process of materials requirements forecasting, with the purpose of better inventory management and supplier integration. The transparency, security, and traceability of the overall process is to be improved, while the process data integrity has to be increased.
B.2	Standardization of the handling and the tools used in forecasting.
B.3	Secure know how (often hidden in dispersed spreadsheets' formulae) on forecasting.
B.4	What-if simulation facility.
B.5	Traceability of information flow.
B.6	Secure and stable interface management between systems involved.
B.7	Broad user accessibility with distributed roles and standard protection procedures.
B.8	Feedback loop for confirmation of imported sales data, done by sales responsible, after “snapshot” import.

*Table 1: Business Requirements*

## 2.6 Technical requirements for the ADAM system

Besides the business issues behind the new forecasting system, the IT infrastructure also imposes specific requirements for software development (Table 2).

ID	Technical Requirements
T.1	Standardization of GUI (buttons, menus), adherence to corporate style guide.
T.2	Solution is to be implemented on existing platforms (SAP R/3, Lotus Notes). No modifications or extensions should be made in SAP R/3 (no ABAP programming).
T.3	Software design ready for operation under Lotus Notes 4.6, SAP R/3 3.0E and 4.6B.
T.4	Access security concept, using Lotus Notes security components.

*Table 2: Technical requirements*



## 3 Technical background

### 3.1 SAP R/3

#### 3.1.1 The SAP R/3 package

As Ovanesyan (1999) describes, *“SAP R/3 is a self-sufficient, real-time, multi-user, client-server and multi-platform modular ERP information system that emulates virtually every aspect of a modern enterprise. SAP R/3 allows a certain degree of customization while enforcing a variety of strict data integrity driven rules. It has its own internal programming language (ABAP/4) and uses a variety of database back-ends (SQL Server, Oracle, Informix, etc.). SAP R/3 implemented event-chains and data objects well before event driven programming became a reality.”*<sup>6</sup>

#### 3.1.2 Client-Server model

The R/3 architecture is based on a software-oriented, multi-tier, client/server principle.<sup>7</sup> Although SAP can be implemented in a centralized (one-tier) configuration, it usually utilizes a distributed architecture, depending on customer requirements. The three levels implemented by SAP are the *presentation, application, and database* layers.

In the classical tree-tier R/3 configuration, users are entering and retrieving data in a dialog with the presentation clients (SAP GUI). The data entered is passed to the application engine, which is responsible for one of the most distinctive functions of the R/3 system: To guarantee the business- and object-related data integrity of the system. The application layer maintains the dialog with the user via the presentation layer, on one side, and on the other, forwards the validated data to the database system.

Figure 4 shows that, besides the classical direct user access through the SAP GUI, alternative forms of interaction are possible. In the example, an application "ExtApp 1" may insert data into tables dTab2, dTab4, dTab5, and dTab6, using the "Application process 2", which enforces the same integrity rules on incoming data as the "Application process 1" imposes on data entered through the SAP GUI.

---

<sup>6</sup> Ovanesyan (1999), p. 12.

<sup>7</sup> SAP (1994), pp. 2/1-2/25.

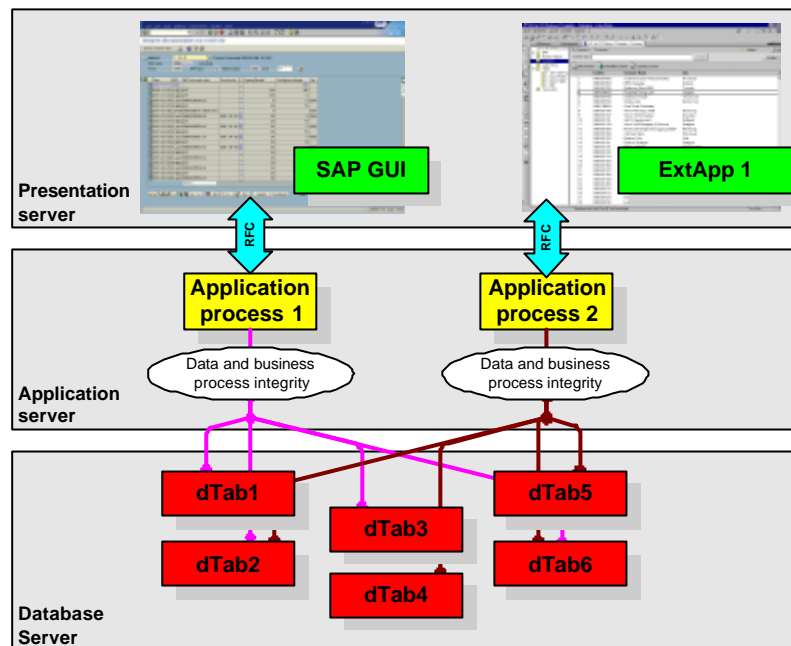


Figure 4: Three tier concept and data handling in SAP

### 3.1.3 Interfacing SAP R/3 with external systems

SAP offers several interfaces for connecting R/3 to external systems, and on different levels; Moser (1999) depicts an overview of the different possibilities.<sup>8</sup> Technologies like EDI or IDoc communication primitives were, because of their high complexity and message based character, somewhat out of the reach of this project, whereas OLE's stability still raised concerns. In the context of this paper we will therefore examine only the details of the RFC and the BAPI technology, which are currently the most used approaches for the ADAM type of application integration.

#### 3.1.3.1 RFC technology

RFCs (Remote Function Calls) are used for communications between two independent SAP systems, or for communications between an SAP system and a non-SAP system, such as an external application. They can also be used for communications between SAP function modules on the same system.<sup>9</sup> From a technical point of view, RFCs are using the TCP/IP protocol stack in client/server, or SNA LU6.2 in mainframe environments. The communication between two systems can be managed synchronously or asynchronously.<sup>10</sup> An external (client) program has to first set up a connection through the local RFC software, then sends its function execution request, and terminates the session with a close command.

<sup>8</sup> Moser (1999), pp. 32-39.

<sup>9</sup> SAP (1999).

<sup>10</sup> Moser (1999), p. 33.

To support developers of external applications for R/3, SAP ships an RFC Software Development Kit (RFCSDK) together with its GUI software. This SDK contains libraries and interfaces for various programming languages, including C, C++, JAVA, and OLE-enabled environments.

On the SAP application side, the RFC function modules are ABAP scripts, which may be called either from other ABAP programs or external applications. Figure 5 shows the implementation of the RFC function RFC\_CUSTOMER\_GET.

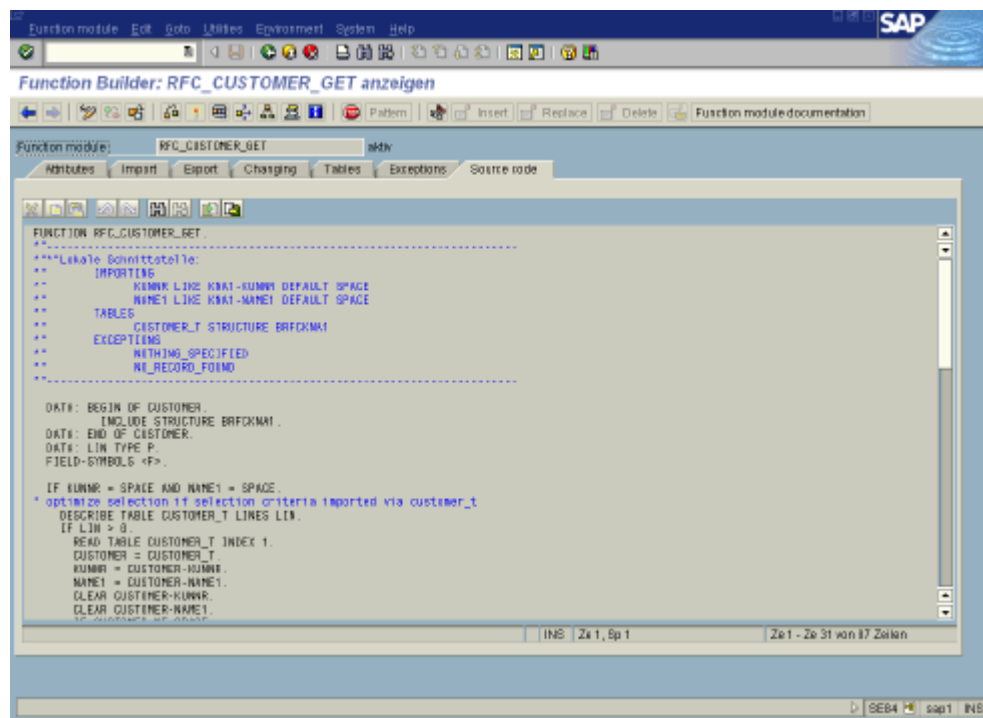


Figure 5: Implementation of the RFC function RFC\_CUSTOMER\_GET<sup>11</sup>

### 3.1.3.2 BAPI technology

SAP has recently introduced object-oriented technology in the R/3 system by making R/3 processes and data available in the form of SAP Business Objects (BO). These software components cover a broad range of R/3 business data that can be accessed through stable and standardized interface methods, which are called Business Application Programming Interfaces (BAPIs).<sup>12</sup>

The Business Objects contain in their kernel information on how to access the data needed within the SAP data model. The data "insert" or "update" operations on Business Objects are strictly protected by object-related (internal) constraints and business rules which determine the allowable range of values the stored data may take.

<sup>11</sup> SAP (2000a).

<sup>12</sup> Moser (1999), p. 54.

Techniques for accessing BAPIs include COM/DCOM (from Microsoft), RFC, or JAVA.<sup>13</sup> The following code excerpt taken from a sample Visual Basic application program shows the ease of use of BAPIs in conjunction with the SAP ActiveX control:<sup>14</sup>

```
'Create a BAPI ActiveX control object
Set oBAPICtrl = CreateObject("SAP.BAPI.1")
'Create a local instance of the SAP Business Object "SalesOrder"
Set boOrder = oBAPICtrl.GetSAPObject("SalesOrder")
'Now call the method of the created object
instanceboOrder.CreateFromData OrderHeaderIn:=oHeader, _
OrderPartners:=oPartners, _
OrderItemsIn:=oItemsIn, _
Return:=oReturn
'Clean up by releasing the SAP Business Object and freeing the BAPI
ActiveX Control object
```

*Code Fragment 1: Accessing BAPI code from Visual Basic*

### 3.1.4 Comparison and selection of RFC and BAPI technologies

Table 3 gives an overview of distinctive features of the RFC and BAPI technologies.

	RFC	BAPI
ISO/OSI level classification <sup>15</sup>	Presentation layer	Application layer
Implementation in SAP R/3 core	ABAP function	ABAP function
Lotus Script library provided for Notes	Yes (SAP LSX)	Yes (SAP LSX)
Accessibility to SAP materials master data (on plant level)	Yes: Through direct table read	No BAPI available

*Table 3: Comparison of RFC and BAPI technologies*

Due to the lack of "out of the box" functionality provided by the current (3.0E and 4.6B) functionality, the BAPI solution could not been considered for the ADAM project. Large companies like, e.g., Swisscom, are building their own BAPIs for certain applications.<sup>16</sup>

<sup>13</sup> SAP (2000b).

<sup>14</sup> SAP (2000b).

<sup>15</sup> Moser (1999), p. 32.

<sup>16</sup> Radding (1999).

Adding BAPIs always means "adding ABAP coding", which would change the R/3 applications portfolio and as such contradict requirement T.2 in Table 2.

## 3.2 Lotus Notes

### 3.2.1 Product idea

The Lotus Notes family is the leading application suite in the groupware market. More than 50 million seats installed worldwide make it the premier application in collaborative office authoring tools.<sup>17</sup>

The Notes concept is based on *distributed document databases* that help users to create and share information within organizations. Documents may be created, displayed, and modified using forms and views. Notes documents are collections of data items, bundled with type and format information, and a layout presentation (form). Among the different data types provided, Rich Text fields are the most powerful elements, providing hypertext functionality: they may contain formatted text, links to other Notes objects (documents, views, or databases), or internet resources. For multiple users working in parallel on a given Notes document, the server software ensures secure operation by using an "intelligent" conflict resolution mechanism. Each database is protected by a specific, role-based access scheme that allows individuals or groups to access documents.

Its offline (disconnected operation) capability, the multi-level security concepts, and the Web-enabled Domino server make Lotus Notes a one-fits-all instrument for intra- and extra-organizational communication.

### 3.2.2 System components<sup>18</sup>

The Notes/Domino family contains the following components:

*Domino Server:* The Domino Server administrates shared databases and handles both the internal and external (Internet) communication. Besides the server operation with Notes clients, a Domino server is able to run POP3, SMTP, and other Internet protocols. When running as a HTTP server, it is both able to dispatch Notes database contents to a browser client or to deliver Notes contents to an external Web server.

*Notes clients:* A Notes client may access both Notes databases and web contents. For the Server administration and the design of Notes databases, specialized client applications are provided (Domino Administrator and Domino Designer, respectively).

---

<sup>17</sup> Hase (2000).

<sup>18</sup> Fricke (1999).

*Web clients:* When using the Domino server as a Web engine, most of the Notes database content may be accessed via a Web browser (yet, Web access was not a criteria for the ADAM software).

*Notes standard applications:* E-mail will remain one of the most commonly used features of any groupware application. Notes handles E-mails the same way as any Notes document, stores them in a user-specific (mail) database, and supports the usage of standard elements like graphics, tables, or hyperlinks to other Notes database items.

Much of the same applies to the *Task Management and Calendar* functions, that are both closely integrated with the E-mail application.

### 3.2.3 Application programming for Lotus Notes

Notes provides users with several programming languages for developing customized applications. Besides the recently introduced, Internet-driven languages like Java and JavaScript, Notes automation may be implemented using:

- (Notes) Formula language (@Functions),
- (Notes) Command language (@Commands), and
- LotusScript

The Formula and Command languages are especially helpful and straightforward for adding GUI functionality (to open documents, trigger import procedures, etc.), and have been heavily used in the ADAM user interface development. LotusScript is a fully fledged, object-oriented programming environment, also supported by other Lotus products, including a full-text editor with syntax highlight, debugger, and data logging functionality.

LotusScript is an embedded, Basic scripting language with a powerful set of language extensions that support object-oriented application development.<sup>19</sup> The interface to Notes is through predefined object classes. Notes compiles and loads user scripts at runtime, if needed, and automatically includes Notes class definitions.<sup>20</sup>

The most important aspects of LotusScript are:<sup>21</sup>

- It is a superset of the Basic language, thus easy to learn for Visual Basic programmers.
- Cross-platform capability: LotusScript code runs unaltered on several platforms, such as Windows, MacOS, or UNIX systems.

---

<sup>19</sup> Lotus (1999b).

<sup>20</sup> Thompson (1999), p. 2.

<sup>21</sup> Lotus (1999b).

- Object-oriented and event-driven: LotusScript code can use object classes provided by other Lotus (or external) libraries. LotusScript classes support single inheritance, constructors/destructors and method overriding. This functionality allows users to take advantage of object-oriented programming, and to rapidly prototype their own custom business objects. Scripts may be connected to events, typically those triggered by user interaction with the GUI, such as "button pressed" actions.
- Library functionality: Scripts can be grouped into libraries and referenced to with the `use` directive.
- LotusScript Extensions (LSX): LotusScript allows users to create their own classes and objects, called LotusScript extensions (LSXs). Specialized scripts from third party providers are available.

### 3.2.4 Interfacing Lotus Notes with ERP systems

After years of development efforts on both SAP and Lotus sides, a multitude of integration tools is today available.<sup>22</sup>

From the R/3 report extraction tool ("Lotus Connection for R/3") or the high level, high performance data integration engines like the "Lotus Enterprise Integrator" (LEI) down to E-mail and Workflow interconnection tools, solutions are offered for all desired levels of integration.

### 3.2.5 The LotusScript extension library for SAP R/3 (SAP LSX)

The LotusScript extension for R/3 enables Domino applications to both send and receive R/3 data, i.e., to read and/or update data in an SAP R/3 system.<sup>23</sup> Data exchange can take place either synchronously or asynchronously via a Domino server agent invoked by the LSX. Calls can also be invoked from a client.<sup>24</sup>

The LSX module places an object-oriented wrapper around SAP's Remote Function Call Software Development Kit (RFCSDK, see 3.1.3.1), which allows programmers to access the RFC and BAPI functionality of the SAP business objects. The SAP LSX object classes support the application development for Notes and Domino Web applications, but are too limited when it comes to E-mail and Workflow integration. The software product "LSX for SAP R/3" is available free of charge on the R/3 GUI CD, starting from Release 3.1G.<sup>25</sup>

Specifically, the SAP LSX consists of two DLLs, that are either installed automatically with the SAP GUI or may be added manually with the Registry Editor (under WinNT/2000).

---

<sup>22</sup> Moser (1999), pp. 43-46.

<sup>23</sup> Moser (1999), p. 50.

<sup>24</sup> Morris (1999).

<sup>25</sup> Moser (1999), p. 88.

### 3.2.5.1 Connecting a LotusScript application to SAP LSX

The connection to the SAP LSX functionality must be initialized by a USELSX " \*rfc " statement. After this initialization, the entire SAP LSX object model is accessible.

### 3.2.5.2 The SAP LSX object model

The SAP LSX object model is shown in Figure 6.

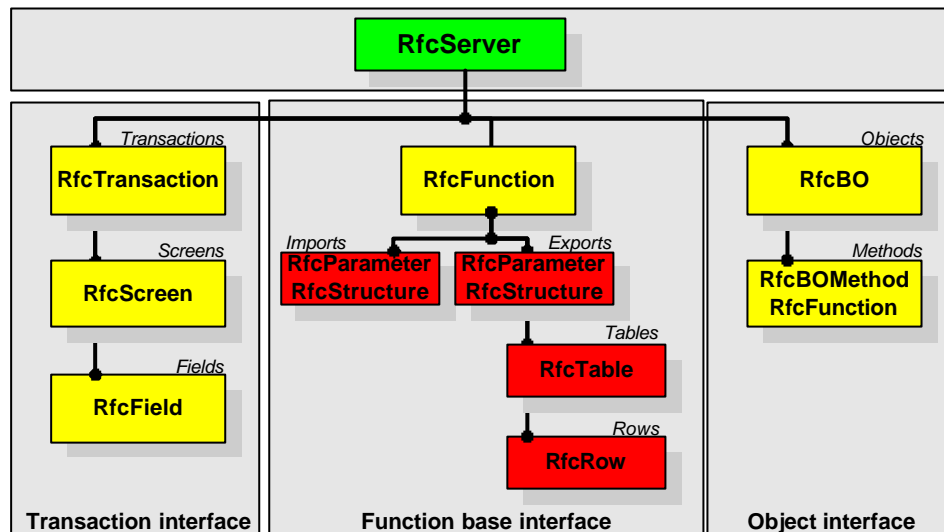


Figure 6: The SAP LSX object model<sup>26</sup>

The SAP LSX comprises three levels of functionality, of which only one will be used in the ADAM context:

- The "Transaction interface" allows the programmer to call SAP transactions, to set GUI screen fields to given values, and to evaluate the parameters returned by the calling program.
- The "Function base interface" covers the central RFC access, where parameters are passed to the called R/3 function (through "Exports"), and "Imports" are expected as return values from the RFC. Additionally, bulk data may be transferred using the "Tables" parameter which contains data transferred in rows and columns.
- A third possibility consists in using the "Object interface" which supports a classic object-oriented access: SAP Business Objects may be accessed directly, using their business object (BAPI) methods.

<sup>26</sup> Lotus (1999a).



### 3.2.5.3 Characteristics of the "function base interface"

Table 4 describes the objects of the "function base interface".<sup>27</sup>

Class/Method/Property	Description of class and its main items
<b>RfcServer</b> <i>Properties:</i> <ul style="list-style-type: none"> <li>.Destination</li> <li>.HostName</li> <li>.System</li> <li>.GatewayHost</li> <li>.GatewayService</li> <li>.TraceLevel</li> <li>.Client</li> <li>.User</li> <li>.Language</li> <li>.Password</li> <li>.Message</li> <li>.SAPVersion</li> </ul> <i>Methods:</i> <ul style="list-style-type: none"> <li>.Load</li> <li>.Logoff</li> <li>.Logon</li> <li>.Save</li> </ul>	<p>The RFC Server describes the basic session object needed to access an SAP R/3 system.</p> <p>A successful logon sequence (i.e., execution of the <code>RfcServer.Logon</code> method) requires that the connection attributes are set, similarly to the SAP R/3 GUI login parameters.</p> <p>The <code>SAPVersion</code> property returns the SAP R/3 server software version, after successful connection.</p> <p>Several possibilities exist in order to <code>.Load</code> or <code>.Save</code> connection parameters from an <code>.ini</code> file.</p>
<b>RfcFunction</b> <i>Properties:</i> <ul style="list-style-type: none"> <li>.Exports</li> <li>.Imports</li> <li>.Message</li> <li>.Name</li> <li>.Parent</li> <li>.Tables</li> </ul> <i>Methods:</i> <ul style="list-style-type: none"> <li>.Call</li> </ul>	<p>The <code>RfcFunction</code> class is used to manage RFC functions and their context. When the function object is created, all appropriate Imports/Exports parameters (see <code>RfcParameter</code> class) as well as the <code>Tables</code> objects (see <code>RfcTable</code> class) will be created automatically. The Lotus Notes Exports parameters will be used as imports on the SAP side, and vice versa; the <code>Tables</code> property contains SAP table data. The <code>Message</code> item contains a message, if an error has occurred during the <code>Call</code> of the RFC function named <code>Name</code>.</p>
<b>RfcParameter</b> <i>Properties:</i> <ul style="list-style-type: none"> <li>.IsStructure</li> <li>.Name</li> <li>.Parent</li> <li>.Type</li> <li>.Value</li> </ul> <i>Methods:</i> <ul style="list-style-type: none"> <li>.Clear</li> </ul>	<p>The <code>RfcParameter</code> class is used to edit Import and Export parameters for RFC functions (see <code>RfcFunction.Imports</code> and <code>RfcFunction.Exports</code>).</p> <p>The <code>Clear</code> method resets a <code>Value</code> to "blank" status.</p>

<sup>27</sup> Lotus (1999a).

Class/Method/Property	Description of class and its main items
<b>RfcTable</b> <i>Properties:</i> .ColumnCount .ColumnName .Name .Parent .RowCount .Rows .SetCell .Table .Width <i>Methods:</i> .GetCell	<p>The RfcTable object contains columns and rows. Depending on its functional context a table may serve as input or output for an RFC function to be executed. Whenever a new RFC function object is created, all tables required will be initialized automatically.</p> <p>The SetCell and GetCell methods allow to modify/read the content of a specified attribute (specified by its row and column indices) of the table object.</p>
<b>RfcRow</b> <i>Methods:</i> .GetCell	<p>The row object supports the access of individual rows within an RfcTable. The row class is "auto-expanded" so that individual column values may be accessed directly by their column name with GetCell.</p>

*Table 4: Extract from the SAP LSX object model (function base interface)*

#### 3.2.5.4 Sample code for system logon and RFC execution

The following code shows the usage of the SAP LSX for logging onto an SAP system:

```
Dim objServer as New RFCserver
'setting the connection parameters for SAP HCC installation
objServer.HostName = "sap.fancy.com"
objServer.System = "XON"
objServer.Client = "600"
objServer.User = "SMITH"
objServer.Language = "D"
objServer.Password = "JOHN"

'logon ...
If objServer.Logon() <> True Then
    MsgBox "Logon error", 0, objServer.Message
End If
```

*Code fragment 2: Logon to the SAP system*

The next example shows a sequence on how to prepare and execute an RFC (without logon/logoff handling):

```
Dim objServer As RfcServer
Dim RfcCustomerGet As RfcFunction
Dim tblCustomer_t As RFCtable

' log on ...

Set RfcCustomerGet = New RfcFunction(objServer, "RFC_CUSTOMER_GET")
```

```

RfcCustomerGet.Exports("NAME1").Value = "*"
RfcCustomerGet.Exports("KUNNR").Value = "*"

If RfcCustomerGet.Call = True Then

    ' Bind the table object to the function's interface table
    ' "CUSTOMER_T" and scan all rows contained in the table

    Set tblCustomer_t = RfcCustomerGet.Tables("CUSTOMER_T")
    ' autoexpand feature:
    Set tblCustomer_t = RfcCustomerGet.CUSTOMER_T

    ForAll rowCustomer in tblCustomer_t
        MsgBox rowCustomer.KUNNR & " - " rowCustomer.NAME1
    End Forall

    ' log off ...

End If

```

*Code fragment 3: Calling RFC\_CUSTOMER\_GET from LotusScript<sup>28</sup>*

The task of this fragment is to display all customer entries present in the SAP database, one after the other. The example should be seen in conjunction with section 3.1.3.1, where the structure of the corresponding (ABAP) RFC program is shown (Figure 5).

### 3.3 Data access through direct table data retrieval

After due consideration, the ADAM project team decided in an early stage that the interfacing had to be implemented using the SAP LSX approach, which is, for the time being, the most popular approach.<sup>29</sup>

For security reasons, the direction of data flow goes only in the direction "SAP R/3 towards Lotus Notes", i.e., the applications developed are allowed read only accesses on R/3 tables and evaluate them with Lotus scripts. This access mode is equivalent to the ABAP Query tool intended for SAP "power user" report programming.<sup>30</sup>

Besides the session control functions (opening and closing connections to the R/3 system), the only RFC function used is the RFC\_READ\_TABLE. Appendix A shows the corresponding ABAP program code.

The RFC\_READ\_TABLE function allows external access to R/3 tables via RFC.<sup>31</sup> Basically, the function block executes an Open SQL statement on the R/3 database,<sup>32</sup> depending on the input data provided by the calling application:

<sup>28</sup> Lotus (1999c).

<sup>29</sup> Huffman/O'Neill/Worthington (2000).

<sup>30</sup> SAP (1999).

<sup>31</sup> SAP (2000a). Transaction code SE84; function block RFC\_READ\_TABLE.

---

```
SELECT * FROM (QUERY_TABLE) INTO WORK WHERE (OPTIONS);
```

---

RFC\_READ\_TABLE may retrieve both structure information and data content of tables residing in the SAP R/3 repository. The output of the function is passed through the DATA parameter, which carries a reference to a complex table. Each row of the (single column) table DATA corresponds to a row of the R/3 table, while the column values are returned as strings, separated by a delimiting character.

The FIELDS parameter contains the list of table attributes to be retrieved. The OPTIONS parameter specifies the SQL "where" conditions, which may contain SQL wildcard operators (% , \_ , etc.). The amount of data returned per row is limited to 512 bytes.

RFC\_READ\_TABLE forwards the OPTIONS without any checking to ABAP/4-SQL; on syntax violation, the function returns an EXCEPTION.

---

<sup>32</sup> Mende (1998), pp. 103-109.

## 4 The ADAM concept and the SAP interfaces

### 4.1 ADAM physical system layout

In order to fulfill the requirement B.7 in Table 2, the functionality of the entire system has been distributed over more than one physical Notes databases:

- `ADAM.nsf` is the central data store, which contains all imported material and sales data, as well as the simulation and requirements calculations results. The sales staff ("SalesResponsible") and any other interested person ("Viewer") may be granted read access, the ADAM super user ("Operator") will have *execution*, *change*, and *delete* permission.
- `ADAM-A.nsf` is the system archive (for performance reasons a large number of documents generated during operation may regularly be moved to an archive).
- `ADAM-X.nsf` finally performs the SAP data import. This database contains configuration documents for the data import and the import scripts.

Figure 7 shows the ADAM Notes databases, the three user access roles described, and the SAP LSX integration. The SAP LSX library is to be installed on the PC of the Operator only, all other users may access ADAM data without this software component.

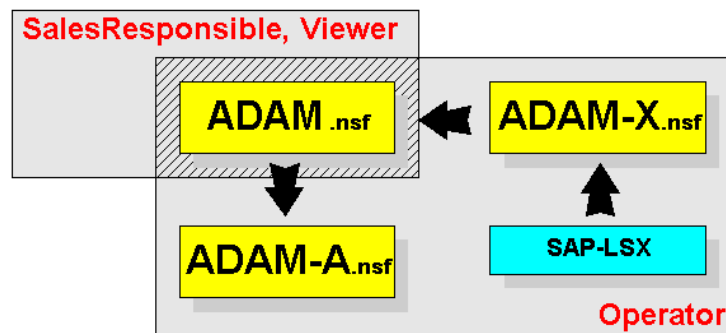


Figure 7: ADAM database solution overview

### 4.2 Logical system architecture

During the system analysis, the ADAM application has conceptually been divided in three subsystems (see Table 5). The *Material* unit cares for the components and quantities needed for material forecast and provides the ADAM system operator with historical data about components consumption and the current state of the requirements plan.

The *Sales* module consists of several interfaces to sales reporting systems, in order to provide snapshots of projects, systems, and components sales forecasts managed by the front end units.

The actual simulation and calculation tasks are performed by the *ADAM Engine* which uses the materials master/historical data on the one and the sales forecast on the other side to build a consolidated, per component requirements forecast.

Subsystem	Tasks	Specific documents
<b>ADAM Material</b>	<ul style="list-style-type: none"> <li>Set up the import options and filters for components</li> </ul>	<ul style="list-style-type: none"> <li>Configuration settings</li> </ul>
	<ul style="list-style-type: none"> <li>Import of components (material) master data</li> </ul>	<ul style="list-style-type: none"> <li>Components data</li> </ul>
	<ul style="list-style-type: none"> <li>Import of supplier information, related to components procurement</li> </ul>	<ul style="list-style-type: none"> <li>Supplier info</li> </ul>
	<ul style="list-style-type: none"> <li>System code to parts list matching.</li> </ul>	<ul style="list-style-type: none"> <li>Matching documents</li> </ul>
	<ul style="list-style-type: none"> <li>Import of bill of materials (BOM) components</li> </ul>	<ul style="list-style-type: none"> <li>BOM structure information</li> </ul>
	<ul style="list-style-type: none"> <li>Import of currently planned requirements of components</li> </ul>	<ul style="list-style-type: none"> <li>Requirements</li> </ul>
	<ul style="list-style-type: none"> <li>Import of components' consumption and replenishment history.</li> </ul>	<ul style="list-style-type: none"> <li>Materials movements trace</li> </ul>
<b>ADAM Sales</b>	<ul style="list-style-type: none"> <li>Define sources for sales input</li> </ul>	<ul style="list-style-type: none"> <li>Source configuration</li> </ul>
	<ul style="list-style-type: none"> <li>Import of sales information: Projects, systems, and components sales</li> </ul>	<ul style="list-style-type: none"> <li>Projects, systems, and components sales</li> </ul>
<b>ADAM Engine</b>	<ul style="list-style-type: none"> <li>Calculus of materials requirements plan</li> </ul>	<ul style="list-style-type: none"> <li>Forecast data</li> </ul>

Table 5: The ADAM subsystems, their tasks and related documents

### 4.3 ADAM Material subsystem architecture

As the results of the requirements consolidation remain in the ADAM system and will not automatically be forwarded to the SAP R/3 database, the only R/3 interface is located in the *ADAM Material* module. The five interfaces created for this purpose are displayed on Figure 8 ("SAP / AM1" through "SAP / AM5").

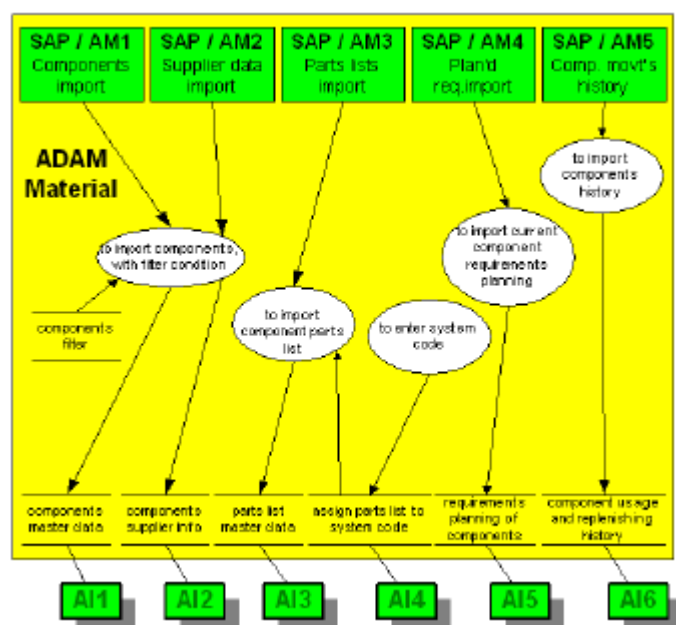


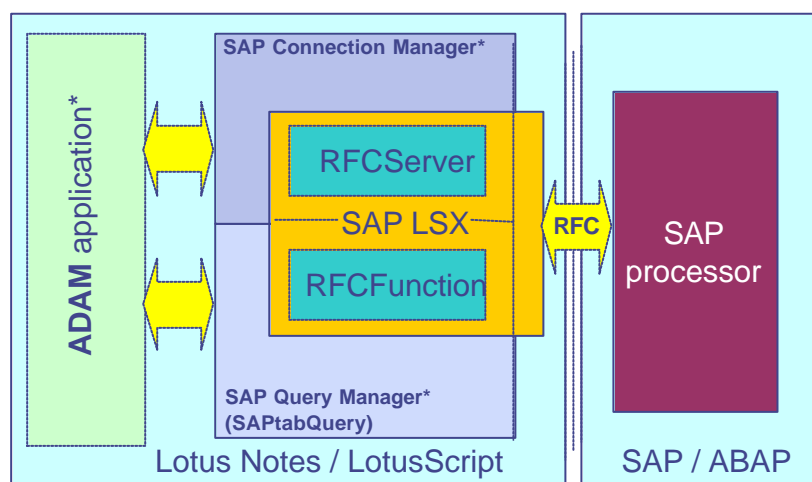
Figure 8: The ADAM Material subsystem data flow

The stores shown in this Data Flow Diagram represent Notes views displaying documents, the elliptic process elements stand for LotusScript procedures (the "processors").

## 4.4 Details of the SAP interface concept

### 4.4.1 The ADAM software support modules

Two functional units, one for managing the session information and the other for supporting SAP queries, have been built on to top of the SAP LSX library (see Figure 9). They free the ADAM applications from individually handling session objects and support the management of several queries in parallel.



\*implemented for ADAM project

Figure 9: ADAM overall structure and software support modules

#### 4.4.2 The SAP Connection Manager

To support ADAM applications with opening and shutting down connections with the SAP processor, a function module named SAP Connection Manager has been created. The two functions implemented, `SAPOpenConnection` and `SAPCloseConnection`, encapsulate the `RFCServer` object handling provided by the SAP LSX module. Figure 10 shows the call sequencing through the different interface levels.

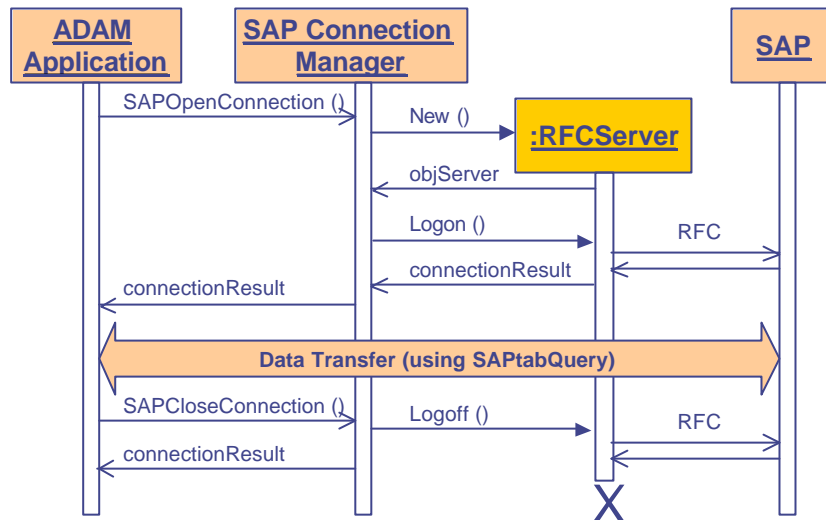


Figure 10: Sequence diagram for the SAP Connection Manager functions

#### 4.4.3 The SAP Query Manager (SAPtabQuery wrapper)

To facilitate the management of variables and RFC method calls needed every time an SAP table has to be accessed, a specialized object class `SAPtabQuery` has been developed, which encapsulates the functionality and data structures needed for data retrieval (see Appendix C).

The following example shows the usage of `SAPtabQuery` object instances (code fragment 4; exception handling only partially shown).

```

'Demo function: Search materials master short text for material pMATNR
Function DemoMAKT (fMATNR as String, pMAKTX As String) as Integer
    Dim intRow As Integer
    Dim tResStr As String
    Dim tSAPtabQuery As SAPtabQuery

    'Create new SAPtabQuery object (constructor call)
    Set tSAPtabQuery = New SAPtabQuery ("MAKT")
  
```



```

'Specify fields to be retrieved (MAKT-SPRAS, MAKT-MAKTX)
tSAPtabQuery.AppendFieldItem ("SPRAS")
tSAPtabQuery.AppendFieldItem ("MAKTX")

'Specify WHERE condition of selection
tSAPtabQuery.AppendOptionItem ("MATNR = '" & fMATNR & "'")

'Now let RFC execute 'select SPRAS, MAKTX from MAKT where MATNR = "fMATNR"'
If tSAPtabQuery.DoCall () = False Then
    '(error handling ...)
    SAPGetMAKTdet = False
Else
    'Process number of rows to be imported
    pMAKTX = ""
    SAPGetMAKTdet = True
    For intRow = 1 To tSAPtabQuery.GetRowCount()
        tResStr = tSAPtabQuery.GetResultItem (intRow, 1)
        If tResStr = "D" Then
            pMAKTX = tSAPtabQuery.GetResultItem (intRow, 2)
            Exit For
        End If
    Next
End If
End Function

```

*Code fragment 4: Application of the SAPtabQuery RFC function wrapper*

Upon execution of DoCall, the parameters previously collected by the tSAPtabQuery wrapper (during the New constructor, the AppendFieldItem and AppendOptionItem method calls) are passed to the RFC interface using the SAP LSX functions.

Figure 11 shows the data items of a sample tSAPtabQuery object instance *before* and *after* an RFC execution. Visibly, the DATA table now contains the result of the selection: a single cell with one column, containing 2 result strings separated by the delimiter ";". The method GetResultItem now extracts the string specified by the "virtual column index", in the code example the string found is "SCREW".

EXPORT	
QUERY_TABLE	"MAKT"
DELIMITER	"."
NO_DATA	" "
ROWSKIPS	0
ROWCOUNT	0

TABLES	
FIELDS	"SPRAS"
	"MAKTX"
OPTIONS	"MATNR = 'xyz'"
DATA	

SAP LSX

Before DoCall ( )

EXPORT	
QUERY_TABLE	"MAKT"
DELIMITER	"."
NO_DATA	" "
ROWSKIPS	0
ROWCOUNT	0

TABLES	
FIELDS	"SPRAS"
	"MAKTX"
OPTIONS	"MATNR = 'xyz'"
DATA	"E;SCREW"

SAP LSX

After DoCall ( )

Figure 11: SAP LSX Object data before and after RFC

## 4.5 Example of data handling in ADAM

### 4.5.1 Design of documents containing imported data

The materials master data are read from the corresponding SAP table and processed in order to be stored in a Notes document in the ADAM database. The design of a part of such a document is shown in Figure 12 (excerpt from ADAM form "fmMat001"):

DISPO / MRP view	
Safety Stock (EISBE):	MARC_EISBE #
Procurement Type (BESKZ):	MARC_BESKZ T
MRP Controller (DISPO):	MARC_DISPO T
MM/PP Status (MMSTA):	MARC_MMSTA T
Production stock (LGPRO):	MARC_LGPRO T
External supply stock (LGFSB):	MARC_LGFSB T
Minimum order size (BSTMI):	MARC_BSTMI T

Value = 101

Figure 12: Materials master data form

Example: After the import of values from the SAP table MARC through the different interfaces, the field MARC\_DISPO (meaning "MRP Controller", of type "Text") may contain the value '101', and will be stored together with other data retrieved from SAP R/3 in one document per material master (e.g., having the component ID 'xyz'). Figure 13 shows a more complete extract from such a materials master data document, after data import:

<b>Identification</b>	
Component ID (MATNR) :	xyz
Component Class (MATKL) :	AJM
Component Short Text (MAKTX) :	SCREW
Component Plant (WERKS) :	1000
Main stock (LGORT) :	0001
<b>Creation Status</b>	
Master status (PSTAT) :	EDPG
<b>Procurement</b>	
Supplier (LIFNR) :	0000001000
Supplier Short Text (NAME1) :	C.E.B. BERLIN
Mail address (INTAD) :	
Replenishing Unit (EKGRP) :	301
<b>DISPO / MRP view</b>	
Safety Stock (EISBE) :	0
Procurement Type (BESKZ) :	F
MRP Controller (DISPO) :	101
MM/PP Status (MMSTA) :	
Production stock (LGPRO) :	0001
External supply stock (LGFSB) :	
Minimum order size (BSTMI) :	0.000

Figure 13: Sample materials master data document

4.5.2 Import filter

The component data import may be controlled by the following filter document ("fmMat006"), in order to process only a selection of the entire SAP R/3 material data:

Material (MATNR) :	P%
<i>May contain SQL wildcards: "%" =&gt; 'any count of anything' "_" =&gt; '1 occurrence of anything'</i>	
MRP Controller (DISP) :	101
Purchasing Group (EKG) :	
Plant (WERKS) :	1000

Figure 14: Materials import filter configuration

The value 'P%' (which contains a SQL wildcard, i.e. a placeholder for any material identification beginning with a "P") of the parameter "Material (MATNR)" will be passed as fMATNR value to the Open SQL queries specified in Appendix B:

Select MATNR, WERKS, PSTAT, MMSTA, BESKZ, EISBE, LGPRO, LGFSB, BSTMI from  
MARC where MATNR like **fMATNR** and DISPO like fDISPO and EKGRP like fEKGRP  
and WERKS like fWERKS;

## **5 SAP LSX: Observations and conclusions**

### **5.1 The SAP LSX concept and software engineering**

The SAP LSX low level interface using RFC technology fits perfectly into the requirements of the ADAM project: high data volume transfer capability in batch mode (peaks during materials master data import/synchronization), support for data storage in Notes documents, and a perfect integration in the Lotus Notes/LotusScript environment. Compared to extensive ABAP/4 programming on the SAP R/3 side, programming in LotusScript and using the SAP LSX library is certainly the easier and better alternative.

The debugging features of the SAP LSX offered for a mixed-platform application like the ADAM software are sufficient. The SAP ABAP debugging mode may automatically be launched with the execution of an SAP LSX-enabled script; the latter may be controlled step-by-step with the integrated Lotus Notes debugger. The SAP RFC trace collects profound information about the call data flow on the SAP LSX interface. Finally, the Lotus Notes standard logging services enable the programmer to implement checkpoints and quickly detect erroneous program states.

From a programmer's view, the LotusScript programming language itself offers many interesting features that beneficially support the manipulation of complex data objects (or collections) like those returned by the SAP LSX functions.

### **5.2 Maturity and reliability**

The SAP LSX solution has now been commercially available for several years. Against all fears, the stability and reliability of the LSX software (when used by LotusScript applications) is surprisingly high. Thousands of materials master data sets have been transferred without error in our ADAM test installation, without any visible destructive effect of the SAP LSX.

### **5.3 Sensibility to the evolution of software products**

Due to the low-level character of the access software, and the fact that data models of large IT systems like SAP's R/3 are forced to remain compatible for years, the ADAM software has been successfully used with different versions of R/3 (3.0E, 3.1H, and 4.6B), without any change in the data import routines. With forthcoming releases of the SAP R/3 and Lotus Notes applications, however, the ADAM software will have to undergo further testing.

### **5.4 Data access**

In the era of the object oriented "paradigm" and technologies like BAPIs, a direct, table-focused data access may appear out-of-date. But, as mentioned earlier, and due to the lack of functionality of the "Material" BAPI, no standard Business Object could be used in the ADAM

project, and custom development of BAPIs was not a feasible alternative. BAPI development always means: extension of the current SAP R/3 application portfolio, i.e. ABAP programming, with all development risks and costly consequences for system maintenance.

The approach of retrieving data from SAP tables using the `RFC_READ_TABLE` is also very similar to the still "popular" and actively marketed *ABAP Query* programming. Both methods suffer from the same "weaknesses": The application developer has to possess a *profound* knowledge of the relevant sections of the SAP R/3 data model. The task of, e.g., finding the right table and joins to extract consolidated data for the monthly consumption of a material, may be regarded as a small "research" project! For non-German speaking people, the analysis of the data model and its attributes may be further hardened by the fact that attribute names are mainly abbreviated German terms.

Finally, once all relevant R/3 tables are found and the appropriate inner joins already in mind, the programmer has to face the fact that not all attributes of a table may qualify to be used in conditions of SQL "where"-clauses.

## 5.5 Conclusions

In the perspective of a (foreseeable) long-term coexistence of Groupware (like Lotus Notes) and ERP systems (e.g., SAP R/3), and an increasing demand for tighter integration of these applications on the users' desktops, the interest in solutions of the "ADAM style" will certainly persist.

The SAP LSX solution provides Lotus Notes programmers with a cost-efficient, pragmatic development tool that grants low-level access to a large part of the SAP R/3 database and functionality. Powerful, reliable, ERP-enabled applications can be developed and deployed with a very short lead time, using Lotus Notes.

Yet, in order to access advanced business data, a thorough understanding of the relevant parts of the huge and complex SAP data model is an absolute must for any developer.

It finally remains open to see if the promised functionality of Advanced Planning Systems (APS) on one side or comprehensive Enterprise Application Integration (EAI) solutions on the other will eventually make outmoded low-end point-to-point integration solutions as the one described here.

## Appendix A: Documentation of RFC\_READ\_TABLE

### A.1 ABAP/4 Code (taken from SAP R/3 repository)<sup>33</sup>

```

FUNCTION RFC_READ_TABLE.
*-----
**"Lokale Schnittstelle:
**  IMPORTING
**      VALUE(QUERY_TABLE) LIKE DD02L-TABNAME
**      VALUE(DELIMITER) LIKE SONV-FLAG DEFAULT SPACE
**      VALUE(NO_DATA) LIKE SONV-FLAG DEFAULT SPACE
**      VALUE(ROWSKIPS) LIKE SOID-ACNT DEFAULT 0
**      VALUE(ROWCOUNT) LIKE SOID-ACNT DEFAULT 0
**  TABLES
**      OPTIONS STRUCTURE RFC_DB_OPT
**      FIELDS STRUCTURE RFC_DB_FLD
**      DATA STRUCTURE TAB512
**  EXCEPTIONS
**      TABLE_NOT_AVAILABLE
**      TABLE_WITHOUT_DATA
**      OPTION_NOT_VALID
**      FIELD_NOT_VALID
**      NOT_AUTHORIZED
**      DATA_BUFFER_EXCEEDED
*-----

CALL FUNCTION 'VIEW_AUTHORITY_CHECK'
  EXPORTING
    VIEW_ACTION          = 'S'
    VIEW_NAME            = QUERY_TABLE
  EXCEPTIONS
    NO_AUTHORITY        = 2
    NO_CLIENTINDEPENDENT_AUTHORITY = 3
    TABLE_NOT_FOUND    = 4.

IF SY-SUBRC = 2 OR SY-SUBRC = 3.
  RAISE NOT_AUTHORIZED.
ELSEIF SY-SUBRC = 1.
  RAISE TABLE_NOT_AVAILABLE.
ENDIF.

*-----
* find out about the structure of QUERY_TABLE
*-----
DATA BEGIN OF TABLE_STRUCTURE OCCURS 10.
  INCLUDE STRUCTURE DNTAB.
DATA END OF TABLE_STRUCTURE.
DATA TABLE_HEADER LIKE X030L.

CALL FUNCTION 'NAMETAB_GET'
  EXPORTING
    TABNAME          = QUERY_TABLE
  IMPORTING
    HEADER           = TABLE_HEADER
  TABLES
    NAMETAB          = TABLE_STRUCTURE
  EXCEPTIONS
    TABLE_HAS_NO_FIELDS = 01
    TABLE_NOT_ACTIV    = 02
    INTERNAL_ERROR      = 03
    NO_TEXTS_FOUND      = 04.

IF SY-SUBRC > 1.
  RAISE TABLE_NOT_AVAILABLE.
ENDIF.
IF SY-SUBRC = 1 OR TABLE_HEADER-TABFORM CN 'TCPV'.
  RAISE TABLE_WITHOUT_DATA.
ENDIF.

*-----
* isolate first field of DATA as output field
* (i.e. allow for changes to structure DATA!)
*-----
DATA LINE_LENGTH TYPE I.
FIELD-SYMBOLS <D>.
ASSIGN COMPONENT 0 OF STRUCTURE DATA TO <D>.
DESCRIBE FIELD <D> LENGTH LINE_LENGTH.

*-----
* if FIELDS are not specified, read all available fields
*-----
DATA NUMBER_OF_FIELDS TYPE I.
DESCRIBE TABLE FIELDS LINES NUMBER_OF_FIELDS.
IF NUMBER_OF_FIELDS = 0.
  LOOP AT TABLE_STRUCTURE.
    MOVE TABLE_STRUCTURE-FIELDNAME TO FIELDS-FIELDNAME.
    APPEND FIELDS.
  ENDLOOP.
ENDIF.

*-----
* for each field which has to be read, copy structure information
* into tables FIELDS_INT (internal use) and FIELDS (output)
*-----
DATA: BEGIN OF FIELDS_INT OCCURS 10.
  TYPE      LIKE TABLE_STRUCTURE-INTTYPE.
  DECIMALS  LIKE TABLE_STRUCTURE-DECIMALS.
  LENGTH_SRC LIKE TABLE_STRUCTURE-INTLEN.
  LENGTH_DST LIKE TABLE_STRUCTURE-DDLEN.
  OFFSET_SRC LIKE TABLE_STRUCTURE-OFFSET,
  OFFSET_DST LIKE TABLE_STRUCTURE-OFFSET,
  END OF FIELDS_INT.
LINE_CURSOR TYPE I.

LINE_CURSOR = 0.
* for each field which has to be read ...
LOOP AT FIELDS.

  READ TABLE TABLE_STRUCTURE WITH KEY FIELDNAME = FIELDS-FIELDNAME.
  IF SY-SUBRC NE 0.
    RAISE FIELD_NOT_VALID.
  ENDIF.

  * compute the place for field contents in DATA rows:
  * if not first field in row, allow space for delimiter
  IF LINE_CURSOR <> 0.
    IF NO_DATA EQ SPACE AND DELIMITER NE SPACE.
      MOVE DELIMITER TO DATA+LINE_CURSOR.
    ENDIF.
    LINE_CURSOR = LINE_CURSOR + STRLEN( DELIMITER ).
  ENDIF.

  * ... copy structure information into tables FIELDS_INT
  * (which is used internally during SELECT) ...
  FIELDS_INT-LENGTH_SRC = TABLE_STRUCTURE-INTLEN.
  FIELDS_INT-LENGTH_DST = TABLE_STRUCTURE-DDLEN.
  FIELDS_INT-OFFSET_SRC = TABLE_STRUCTURE-OFFSET.
  FIELDS_INT-OFFSET_DST = LINE_CURSOR.
  FIELDS_INT-TYPE       = TABLE_STRUCTURE-INTTYPE.
  FIELDS_INT-DECIMALS   = TABLE_STRUCTURE-DECIMALS.
  * compute the place for contents of next field in DATA rows
  LINE_CURSOR = LINE_CURSOR + TABLE_STRUCTURE-DDLEN.
  IF LINE_CURSOR > LINE_LENGTH AND NO_DATA EQ SPACE.
    RAISE DATA_BUFFER_EXCEEDED.
  ENDIF.
  APPEND FIELDS_INT.

  * ... and into table FIELDS (which is output to the caller)
  FIELDS-FIELDTEXT = TABLE_STRUCTURE-FIELDTEXT.
  FIELDS-TYPE      = TABLE_STRUCTURE-INTTYPE.
  FIELDS-LENGTH    = FIELDS_INT-LENGTH_DST.
  FIELDS-LENGTH_SRC = FIELDS_INT-OFFSET_DST.
  MODIFY FIELDS.

ENDLOOP.
* end of loop at FIELDS

*-----
* read data from the database and copy relevant portions into DATA
*-----
* output data only if NO_DATA equals space (otherwise the structure
* information in FIELDS is the only result of the module)
IF NO_DATA EQ SPACE.

  DATA: BEGIN OF WORK, BUFFER(30000), END OF WORK.
  FIELD-SYMBOLS <F>.

  IF ROWCOUNT > 0.
    ROWCOUNT = ROWCOUNT + ROWSKIPS.
  ENDIF.

  SELECT * FROM (QUERY_TABLE) INTO WORK WHERE (OPTIONS).

  IF SY-DBCNT GT ROWSKIPS.

    * copy all relevant fields into DATA (output) table
    LOOP AT FIELDS_INT.
      IF FIELDS_INT-TYPE = 'P'.
        ASSIGN WORK+FIELDS_INT-OFFSET_SRC(FIELDS_INT-LENGTH_SRC)
              TO <F>.
        TYPE      FIELDS_INT-TYPE.
        DECIMALS  FIELDS_INT-DECIMALS.
      ELSE.
        ASSIGN WORK+FIELDS_INT-OFFSET_SRC(FIELDS_INT-LENGTH_SRC)
              TO <F>.
        TYPE      FIELDS_INT-TYPE.
      ENDIF.
      MOVE <F> TO <D>+FIELDS_INT-OFFSET_DST(FIELDS_INT-LENGTH_DST).
    ENDLOOP.
    * end of loop at FIELDS_INT
    APPEND DATA.

    IF ROWCOUNT > 0 AND SY-DBCNT GE ROWCOUNT. EXIT. ENDIF.

  ENDIF.
ENDSELECT.
ENDIF.
ENDFUNCTION.

```

<sup>33</sup> See SAP (2000a).

## A.2 Import Parameters

Name	Default	Optional	Description
QUERY_TABLE			Name of table to be queried
DELIMITER	SPACE	X	Delimiter for separation of column values
NO_DATA	SPACE	X	If this value equals 'X', only the table structure will be retrieved
ROWSKIPS	0	X	(not used)
ROWCOUNT	0	X	Maximum number of rows to be retrieved; if 0, unlimited

## A.3 Export Parameters

Not used.

## A.4 Tables

Name	Description
OPTIONS	Input: WHERE clause of SELECT statement
FIELDS	On Input: Names of fields to be read On Output: Structure of table fields structure
DATA	Retrieved data

## Appendix B: ADAM SAP Interface – SAP table data extraction routines

ID	SAP table	ADAM SAP Interface function	Function description	Open SQL command (simplified)
AM1	MARC	SAPUpdateMatMaster	Import plant-independent materials master data	Select MATNR, WERKS, PSTAT, MMSTA, BESKZ, EISBE, LGPRO, LGFSB, BSTMI from MARC where MATNR like fMATNR and DISPO like fDISPO and EKGRP like fEKGRP and WERKS like fWERKS;
	LFA1	SAPGetLIEFdet	Get supplier name	Select NAME1 from LFA1 where LIFNR = fLIFNR;
	LFB1	SAPGetLIEFdet	Get supplier mail address	Select INTAD from LFB1 where LIFNR = fLIFNR and BUKRS = fBUKRS;
	MAKT	SAPGetMAKTdet	Get material short text	Select MAKTX from MAKT where MATNR = fMATNR and SPRAS = fSPRAS;
	MBEW	SAPGetMBEWdet	Get material valuation data	Select VPRSV, VERPR, STPRS, KOSGR, SPERW, BWTAR from MBEW where MATNR = fMATNR and BWKEY = fBWKEY and BWTAR = fBWTAR;
	MAPL	SAPGetMAPLdet	Check whether material is marked for deletion	Select LOEKZ from MAPL where MATNR = fMATNR and WERKS = fWERKS;
AM2	EINA	SAPGetLiefID	Get supplier code for material	Select LIFNR, RELIF from EINA where MATNR = fMATNR;
AM3	MAST	SAPGetPartsListID	Get existing PL identifier for material	Select STLNR, STLAL from MAST where MATNR = fMATNR and WERKS = fWERKS and STLAN = fSTLAN;



ID	SAP table	ADAM SAP Interface function	Function description	Open SQL command (simplified)
	STKO	SAPVerifySTK0det	Check if PL head is released and valid	Select STKOZ, STLST from STKO where STLNR = fSTLNR and STLAL = fSTLAL;
	STZU	SAPVerifySTZUdet	Check if PL is configurable	Select KBAUS from STZU where STLNR = fSTLNR and STLTY = fSTLTY and STLAN = fSTLAN;
	STAS	SAPOpenPLComp	Get complete PL position ids	Select STLKN, LKENZ, STLTY, STLAL, AENNR from STAS where STLNR = fSTLNR and STLTY = fSTLTY and STLAL = fSTLAL and LKENZ = ' ';
	STPO	SAPGetSTPOdet	Get PL position details	Select IDNRK, POSNR, MEINS, MENGE, SANFE, AENNR from STPO where STLNR = fSTLNR and STLTY = fSTLTY and STLKN = fSTLKN;
AM4	PBED	SAPGetPBEDdet	Get materials primary requirements plan	Select PLNMG, PDATU from PBED where BDZEI = fBDZEI;
AM5	S031	SAPGetMatHistory	Read historical materials movements (in and out)	Select MZUBB, WZUBB, MAGBB, WAGBB, SPMON from S031 where MATNR = fMATNR and WERKS = fWERKS and LGORT = fLGORT;

## Appendix C: The SAPtabQuery function wrapper<sup>34</sup>

```

Class SAPtabQuery
    objRfcReadTable As RFCFunction
    tblFields As RFCTable
    tblOptions As RFCTable
    tblData As RFCTable
    itmFieldNames As NotesItem
    itmFieldValues As NotesItem
    executed As Integer

'*** Constructor
    Sub New (pSAPtab As String)
        'Declare RFC relevant objects
        Set objRfcReadTable = New RFCFunction (objServer, "RFC_READ_TABLE")

        'Assign the "Import" parameters of the function module.
        'Name of table to query
            objRfcReadTable.Exports("QUERY_TABLE").Value = pSAPtab
        'Set dellimiter between column values
            objRfcReadTable.Exports("DELIMITER").Value = ";"
        'Query data, not structure information
            objRfcReadTable.Exports("NO_DATA").Value = " "
            objRfcReadTable.Exports("ROWSKIPS").Value = 0
            objRfcReadTable.Exports("ROWCOUNT").Value = 0
        'Set FIELDS table
            Set tblFields = objRfcReadTable.Tables("FIELDS")
        'Set OPTIONS table
            Set tblOptions = objRfcReadTable.Tables("OPTIONS")
        'Set table object to result table DATA
            Set tblData = objRfcReadTable.Tables("DATA")
            executed = False
            sepChar = ";"
        End If
    End If
End Sub

'*** Set Field items
    Sub AppendFieldItem (pField As String)
        Dim objRow As RfcRow
        Set objRow = tblFields.Rows.InsertRow ()
        Call objRow.SetCell (1 , pField)
    End Sub

```

<sup>34</sup> Exception handling not shown.

```

'*** Set Options items
    Sub AppendOptionItem (pField As String)
        Dim objRow As RfcRow
        Set objRow = tblOptions.Rows.InsertRow ()
        Call objRow.SetCell (1 , pField)
    End Sub

'*** Execute RFC call
    Function DoCall ()
        If executed = False Then
            DoCall = objRfcReadTable.Call ()
            If DoCall = False Then
                Call SAPCloseConnection ()
                (... error handling ...)
            End If
        Else
            'double call = illegal !
        End If
        executed = True
    End Function

'*** Get result message after RFC has been executed
    Function GetMessage ()
        GetMessage = objRfcReadTable.Message
    End Function

'*** Get number of columns returned in the DATA table
    Function GetColumnCount()
        GetColumnCount = tblData.columncount
    End Function

'*** Get number of rows returned in the DATA table
    Function GetRowCount()
        GetRowCount = tblData.Rows.Count
    End Function

'*** Get result item out of DATA table
    Function GetResultItem (pRow As Integer, pCol As Integer) As Variant

        tResStr = tblData.GetCell (pRow, 1)
        (... now follows code to find item pCol in ";"-separated string ...)

    End Function

'*** Destructor (automatically managed by Lotus runtime system)
    Sub Delete
    End Sub

End Class

```

## Abbreviations

Abbreviation	Description
ABAP (/4)	"Advanced Business Application Programming", SAP's proprietary 4 <sup>th</sup> generation programming language.
ADAM	"Automation of <i>Data</i> flow from <i>sAles</i> forecast to <i>Materials</i> requirement planning" ( <i>„Automatisierung des Datenflusses zwischen Absatzvorschau und Materialprognose“</i> , in German)
BAPI	Business Application Programming Interface
BOM	Bill of material
ERP	Enterprise Resource Planning
DLL	Dynamic Link Library (software component)
LSX	LotusScript Extension
MRP	Materials Requirement Planning
RFC	Remote Function Call

## Code fragments

Code Fragment 1: Accessing BAPI code from Visual Basic .....	9
Code fragment 2: Logon to the SAP system .....	15
Code fragment 3: Calling RFC_CUSTOMER_GET from LotusScript .....	16
Code fragment 4: Application of the SAPtabQuery RFC function wrapper.....	22

## Tables

Table 1: Business Requirements.....	5
Table 2: Technical requirements.....	5
Table 3: Comparison of RFC and BAPI technologies .....	9
Table 4: Extract from the SAP LSX object model (function base interface).....	15
Table 5: The ADAM subsystems, their tasks and related documents .....	19

---

## Figures

Figure 1: Manually entered materials requirements in SAP .....	3
Figure 2: Sales Projects overview.....	4
Figure 3: Materials requirements planning with ADAM.....	4
Figure 4: Three tier concept and data handling in SAP .....	7
Figure 5: Implementation of the RFC function RFC_CUSTOMER_GET .....	8
Figure 6: The SAP LSX object model.....	13
Figure 7: ADAM database solution overview.....	18
Figure 8: The ADAM Material subsystem data flow.....	20
Figure 9: ADAM overall structure and software support modules.....	20
Figure 10: Sequence diagram for the SAP Connection Manager functions .....	21
Figure 11: SAP LSX Object data before and after RFC .....	23
Figure 12: Materials master data form.....	23
Figure 13: Sample materials master data document .....	24
Figure 14: Materials import filter configuration.....	24

## References

[Fricke 1999]

Fricke, R., Lotus Notes / Domino R5. Kaarst: bhv Verlags GmbH 1999.

[Hase 2000]

Hase, M., Bringt der Rabe Glück?, INFORMATIONWEEK (21), 2000, pp. 20-27.

[Huffman/O'Neill/Worthington 2000]

Huffman, M., O'Neill, D., Worthington, E., The Domino Effect: Many new avenues of integration between R/3 and Lotus Domino have opened up, but now you must decide which to take. URL: <http://www.intelligenterp.com/feature/oneill.shtml>  
[as of 2001-01-11].

[Lotus 1999a]

Lotus, Help for the Lotus Script Extension for SAP R/3 - Lotus Script Extension 2.0 for SAP R/3 (distributed with SAP GUI Release 4.6B). File name r3lsxdoc.nsf.

[Lotus 1999b]

Lotus, Domino R5 Designer Help.

[Lotus 1999c]

Lotus, Lotus Script Extension for R/3 Sample Database. Release 2.0 for SAP R/3 (distributed with SAP GUI Release 4.6B). File name Sapexv2.nsf.

[Mende 1998]

Mende, U., Softwareentwicklung für R/3 - Data Dictionary, ABAP/4, Schnittstellen. Berlin, Heidelberg, New York: Springer 1998.

[Morris 1999]

Morris, S., The Best of Two Worlds: Integrating SAP/R3 and Lotus Domino for Bidirectional Data Flow and Workflow.

URL:

[http://www.lotus.com/products/eibu\\_knowbase.nsf/4823032b6687ce59852566fe004e9bd3/3db3978e4bcd753f852567a700597807/\\$FILE/bestof2.pdf](http://www.lotus.com/products/eibu_knowbase.nsf/4823032b6687ce59852566fe004e9bd3/3db3978e4bcd753f852567a700597807/$FILE/bestof2.pdf)

[as of 2001-01-05].

[Moser 1999]

Moser, G., SAP R/3 interfacing using BAPIs: A practical guide to working within the SAP business framework. Braunschweig, Wiesbaden: Vieweg 1999.

[Ovanesyan 1999]

Ovanesyan, O., Professional Visual Basic SAP R/3 Programming. Birmingham: Wrox 1999.

[Pender 2000]

Pender, L., Damned, if you do - will integration tools patch the holes left by an unsatisfactory ERP integration ? URL: [http://www.cio.com/archive/091500\\_erp.html](http://www.cio.com/archive/091500_erp.html) [as of 2001-01-09].

[Radding 1999]

Radding, A., More Than An Application.

URL: <http://www.informationweek.com/728/28iuerp4.htm> [as of 2001-01-10].

[Rost 1999]

Rost, C., Zusammenführung zweier Welten - Möglichkeiten der Anbindung zwischen Lotus Notes und SAP R/3, it FOKUS (12), 1999, pp. 70-75.

[Röthlin 2001]

Röthlin, M., Der Beitrag von Groupware-Applikationen zur methodengestützten Abwicklung von studentischen Projektseminaren, in: Lichter, H., Glinz, M. (Hrsg.), Software Engineering im Unterricht der Hochschulen, Heidelberg: dpunkt.verlag 2001, pp. 55 - 64.

[SAP 1994]

SAP, Funktionen im Detail: SAP R/3 Software-Architektur. 1994.

[SAP 1999]

SAP, SAP Library - Release 4.6B. [as of 2001-01-05].

[SAP 2000a]

SAP, SAP Release 4.6B Client Software.

[SAP 2000b]

SAP, A. H., BAPI Introduction and overview.

URL: <http://www.sap.com/solutions/technology/bapis/edu/docu/caalbe/caalbe.htm> [as of 2000-10-16].

[Thompson 1999]

Thompson, W., LotusScript - Accelerated Lotus Study Guide. New York et al.: McGraw-Hill 1999.